IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: **Haugh et al.**	§	
	§	Group Art Unit: **2135**
Serial No. **10/687,258**	§	
	§	Examiner: **Baotran N. To**
Filed: **October 16, 2003**	§	
	§	
For: **Method, Apparatus, and Program for Multiple Simultaneous ACL Formats on a Filesystem**	§	

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

35525
PATENT TRADEMARK OFFICE
CUSTOMER NUMBER

APPEAL BRIEF (37 C.F.R. 41.37)

This brief is in furtherance of the Notice of Appeal, filed in this case on June 29, 2007.

A fee of $500.00 is required for filing an Appeal Brief. Please charge this fee to IBM Corporation Deposit Account No. 09-0447. No additional fees are believed to be necessary. If, however, any additional fees are required, I authorize the Commissioner to charge these fees which may be required to IBM Corporation Deposit Account No. 09-0447. No extension of time is believed to be necessary. If, however, an extension of time is required, the extension is requested, and I authorize the Commissioner to charge any fees for this extension to IBM Corporation Deposit Account No. 09-0447.

REAL PARTY IN INTEREST

The real party in interest in this appeal is the following party: International Business Machines Corporation of Armonk, New York.

RELATED APPEALS AND INTERFERENCES

With respect to other appeals or interferences that will directly affect, or be directly affected by, or have a bearing on the Board's decision in the pending appeal, there are no such appeals or interferences.

STATUS OF CLAIMS

A. TOTAL NUMBER OF CLAIMS IN APPLICATION

Claims in the application are: 1-20.

B. STATUS OF ALL THE CLAIMS IN APPLICATION

1. Claims canceled: None.
2. Claims withdrawn from consideration but not canceled: None.
3. Claims pending: 1-20.
4. Claims allowed: None.
5. Claims rejected: 1-20.
6. Claims objected to: None.

C. CLAIMS ON APPEAL

The claims on appeal are: 1-20.

STATUS OF AMENDMENTS

A First Office Action was mailed December 14, 2006; Response to First Office Action was filed March 14, 2007; a Final Office Action was mailed May 31, 2007; no response was filed to the Final Office Action; a Notice of Appeal was filed June 29, 2007. An amendment after Final Rejection was not filed in this case. Therefore, claims 1-20 on appeal herein are as amended in the Response to First Office Action dated March 14, 2007.

SUMMARY OF CLAIMED SUBJECT MATTER

A. CLAIM 1 - INDEPENDENT

The subject matter of claim 1 is directed to a method for managing access control lists in a filesystem (Specification, p. 8, lines 6-11; p. 21, lines 11-20; and Figure 4, item 410), the method comprising associating two or more access control lists with a given filesystem object in a heterogeneous filesystem (Specification, p. 8, lines 12-15; p. 14, lines 1-23; and Figure 4, item 410), wherein the heterogeneous filesystem comprises two or more differing types of filesystems (Specification, p.8, lines 1-3; p. 14, lines 13-29; and Figure 4, items 412-436), responsive to receiving, from a requestor, a request for an access control list associated with the given filesystem object, determining a filesystem type of the requestor (Specification, p. 8, lines 12-19 and lines 24-30; page 19, lines 7-10; and Figure 7, items 702), and returning an access control list from the two or more access control lists for the given filesystem object matching the filesystem type of the requestor (Specification, p. 15, lines 3-11; page 19, lines 21-23; and Figure 7, item 724).

B. CLAIM 11 - INDEPENDENT

The subject matter of claim 11 is directed to an apparatus for managing access control lists in a filesystem (Specification, p. 8, lines 6-11; and Figure 5), the apparatus comprising a filesystem (Specification, p. 14, lines 5-28; and Figure 4, item 410; Figure 5, item 510), wherein the filesystem is a heterogeneous filesystem (Specification, p. 8, lines 1-11; p. 14, lines 5-28; and Figure 4, item 410; Figure 5, item 510), and wherein the filesystem includes a plurality of differing filesystem types (Specification p.8, lines 1-3; p. 14, lines 5-28) and a plurality of access mechanisms (Specification, p. 14, lines 13-29; and Figure 4, items 412-216; Figure 5, item 510), and wherein each access mechanism of the plurality of access mechanisms is associated with a filesystem type (Specification, p. 14, lines 13-29; and Figure 4, items 412-216; Figure 5, item 510), and a file storage (Specification, p. 8, lines 21-24; page 16, lines 9-10; and Figure 5, item 550), wherein the file storage has stored therein at least one filesystem object and wherein a given filesystem object within the at least one filesystem object has associated therewith two or more access control lists (Specification, p. 8, lines 12-15; page 16, lines 9-10; and Figure 5, items 520-570), wherein the filesystem (Specification, Figure 5, item 510),

responsive to receiving from a requestor a request for an access control list associated with the given filesystem object, determines a filesystem type of the requestor and returns an access control list from the two or more access control lists for the filesystem object matching the filesystem type of the requestor (Specification, p. 8, lines 12-19 and lines 24-30; p. 15, lines 3-11; p. 15, lines 24-30; and Figure 7, items 702 and 724).

C. CLAIM 20 - INDEPENDENT

The subject matter of claim 20 is directed to a computer program product in a computer readable recordable-type medium (Specification, p.22, lines 9-27; and Figure 1, item 100; Figure 2, item 232), for managing access control lists in a filesystem (Specification, p. 8, lines 6-11), the computer program product comprising instructions for associating two or more access control lists with a given filesystem object in a heterogeneous filesystem (Specification, p. 8, lines 12-15; and Figure 4, item 410), wherein the heterogeneous filesystem comprises two or more differing types of filesystems (Specification, p.8, lines 1-3; p. 14, lines 5-29; and Figure 4, item 410), instructions, responsive to receiving from a requestor a request for an access control list associated with the given filesystem object for determining a filesystem type of the requestor (Specification, p. 8, lines 12-19 and lines 24-30; and Figure 7, item 702), and instructions for returning an access control list from the two or more access control lists for the given filesystem object matching the filesystem type of the requestor (Specification, p. 15, lines 3-11; and Figure 7, item 724).

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

The grounds of rejection to review on appeal are as follows:

1. Whether claims 1-3, 8, 10-13, 18 and 20 are obvious over *Gai*, et al., Method and Apparatus for Organizing, Storing and Evaluating Access Control Lists, Patent No. 6,651,096, dated November 18, 2003 (hereinafter "*Gai*") in view of *Bradley*, File System Translators and Methods for Implementing the Same, U.S. Patent No. 6,871,245, dated March 22, 2005 (hereinafter referred to as "*Bradley*") under 35 U.S.C. § 103(a); and

2. Whether claims 4-7, 9, 14-17 and 19 are obvious over *Gai* and *Bradley* and further in view of *Hitz* et al., File Access Control in a Multi-Protocol File Server, U.S. Patent No. 6,457,130, dated September 24, 2002 (hereinafter referred to as "*Hitz*") under 35 U.S.C. § 103(a).

ARGUMENT

A. GROUND OF REJECTION 1 (Claims 1-3, 8, 10-13, 18, and 20)

The first ground of rejection is whether the Examiner failed to state a *prima facie* obviousness rejection under 35 U.S.C. § 103 against claims 1-3, 8, 10-13, 18, and 20 as allegedly being unpatentable over *Gai*, et al., Method and Apparatus for Organizing, Storing and Evaluating Access Control Lists, Patent No. 6,651,096, dated November 18, 2003 (hereinafter "*Gai*") in view of *Bradley*, File System Translators and Methods for Implementing the Same, U.S. Patent No. 6,871,245, dated March 22, 2005 (hereinafter referred to as "*Bradley*"). This rejection is in error and should be overturned.

A.1. Claims 1 and 20

Claim 1 is representative of this grouping of claims. Claim 1 is as follows:

A method for managing access control lists in a filesystem, the method comprising:

associating two or more access control lists with a given filesystem object in a heterogeneous filesystem, wherein the heterogeneous filesystem comprises two or more differing types of filesystems;

responsive to receiving, from a requestor, a request for an access control list associated with the given filesystem object, determining a filesystem type of the requestor; and

returning an access control list from the two or more access control lists for the given filesystem object matching the filesystem type of the requestor.

With respect to claim 1, the Examiner states that:

Regarding claims 1 and 20, Gai discloses a method for managing access control lists in a filesystem (see abstract, "organizing, storing and evaluating access control lists"), the method comprising: associating two or more access control lists (see Fig. 4, elements 416a-416e) with a given filesystem object, (see col. 7 lines 15051);

Gai does not disclose "in a heterogeneous filesystem, wherein the heterogeneous file system comprises two or more differing types of filesystems."

However, Bradley explicitly discloses in a heterogeneous filesystem, wherein the heterogeneous filesystem comprises two or more differing types of filesystems (Abstract, col. 7, lines 30-40 and col. 18, lines 1-24).

Therefore, it would have been obvious at the time the invention was made to a person having ordinary skill in the art to have incorporated Bradley's invention within Gai to include a heterogeneous filesystem, wherein the

heterogeneous filesystem comprises two or more differing types of filesystems. One of ordinary skill in the art would have been motivated to do this because there is a need for a file system that can efficiently grant access to heterogeneous platform (Bradley col. 3, lines 9-10).

Gai and Bradley disclose the limitations of Claims 1 and 20 above. Gai and Bradley further disclose responsive to receiving, from a requestor, a request for an access control list associated with the given filesystem object (Gai see col. 4, lines 26-32; col. 7, lines 24-32), determining a filesystem type of the requestor (Gai see col. 5, lines 13-21; col. 7, lines 29-34; col. 8, lines 9-15); and

returning an access control list from the two or more access control list for the given filesystem object matching the filesystem type of the requestor (see Gai col. 8, lines 14-15, "Once a match is located, the corresponding action is returned and processing stops") and (see Bradley Figures 7 and 9, col. 18, lines 1-25).

Final Office Action, dated May 31, 2007, pp. 4-5.

The Examiner bears the burden of establishing a *prima facie* case of obviousness based on prior art when rejecting claims under 35 U.S.C. § 103. *In re Fritch*, 972 F.2d 1260, 23 U.S.P.Q.2d 1780 (Fed. Cir. 1992). The prior art reference (or references when combined) must teach or suggest all the claim limitations. *In re Royka,* 490 F.2d 981, 180 USPQ 580 (CCPA 1974). In determining obviousness, the scope and content of the prior art are... determined; differences between the prior art and the claims at issue are... ascertained; and the level of ordinary skill in the pertinent art resolved. Against this background the obviousness or non-obviousness of the subject matter is determined. *Graham v. John Deere Co.*, 383 U.S. 1 (1966). Often, it will be necessary for a court to look to interrelated teachings of multiple patents; the effects of demands known to the design community or present in the marketplace; and the background knowledge possessed by a person having ordinary skill in the art, all in order to determine whether there was an apparent reason to combine the known elements in the fashion claimed by the patent at issue. *KSR Int'l. Co. v. Teleflex, Inc.*, No. 04-1350 (U.S. Apr. 30, 2007). Rejections on obviousness grounds cannot be sustained by mere conclusory statements; instead, there must be some articulated reasoning with some rational underpinning to support the legal conclusion of obviousness. *Id.* (citing *In re Kahn*, 441 F.3d 977, 988 (CA Fed. 2006)).

A.1.i. The proposed combination of references, considered as a whole, does not teach or suggest the feature of "associating two or more access control lists with a given filesystem object in a heterogeneous filesystem, wherein the heterogeneous filesystem comprises two or more differing types of filesystems," as in claim 1.

The Examiner failed to state a *Prima facie* obviousness rejection against claim 1 because the proposed combination of references, considered as a whole, does not teach or suggest the feature of "associating two or more access control lists with a given filesystem object, wherein the heterogeneous filesystem comprises two or more differing types of filesystems," as in claim 1. The Examiner asserts otherwise, citing Figure 4, elements 416a-416e of *Gai* as disclosing the feature "associating two or more access control lists." Figure 4 of *Gai* illustrates:

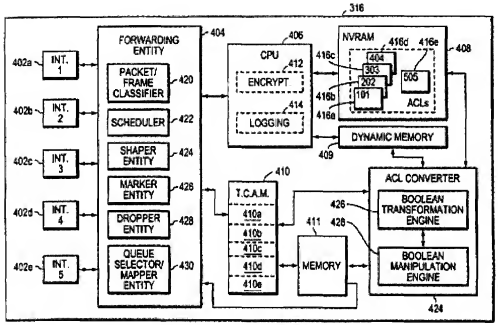


FIG. 4

Figure 4 is an illustration of an intermediate network device operating primarily at the internetwork layer, such as a router. *Gai* states that "FIG. 4 is a partial block diagram of the intermediate network device 316." Gai, column 5, lines 60-61. Gai further explains that "[p]referably, intermediate network device 316 is a layer 3 intermediate device and thus operates up to the network layer of the communication protocol stack implemented within the network 300. " Gai, column 5, lines 23-26. An intermediate network device is further described by *Gai* as follows:

> The term layer 3 intermediate device as used herein are intended broadly to cover any intermediate device operating primarily at the internetwork layer, including, without limitation, routers as defined by Request for Comments (RFC) 1812 from the Internet Engineering Task Force (IETF), intermediate devices that are only partially compliant with RFC 1812, intermediate devices that provide additional

> functionality, such as Virtual Local Area Network (VLAN) support, IEEE 802.1Q support and/or IEEE 802.1D support, etc. Nonetheless, it should be understood that the present invention may also be utilized in whole or in part with other intermediate network devices, such as switches and/or layer 2 intermediate devices, which are also intended to broadly cover any intermediate device operating primarily at the data link layer, including, without limitation, devices that are fully or partially compliant with the IEEE 802.1D MAC Bridge standard and intermediate devices that provide additional functionality, such as Virtual Local Area Network (VLAN) support, IEEE 802.1Q support and/or IEEE 802.1p support, Asynchronous Transfer Mode (ATM) switches, Frame Relay switches, etc.

Gai, column 5, lines 35-54.

Thus, *Gai* makes it abundantly clear that the device depicted in Figure 4 is an intermediate network device for routing and **not a filesystem or a filesystem object.** Therefore, the fact that Figure 4 shows a random access memory (RAM) that includes ACLs 416a-416e stored on the RAM memory, cannot possibly be interpreted as teaching **a given filesystem object in a heterogeneous filesystem that comprises two or more differing types of filesystems** because *Gai* does not even teach, suggest, or mention a filesystem, file system types, or a filesystem object in this or any other section of the reference.

In addition, *Gai* also fails to teach or suggest the step of **associating** two or more access control lists with a given filesystem object, as alleged by the Examiner, because *Gai* is teaching organizing, storing, and evaluating access control lists **in an intermediate network device, such as a router,** and not a filesystem, as shown above. *See also Gai* abstract. Moreover, even if the router of *Gai* could, *arguendo*, be interpreted so broadly as to teach or suggest two or more access control lists in a filesystem, *Gai* only shows the access control lists stored in a RAM memory. *Gai* does not teach or suggest associating two or more of the access control lists with a filesystem object or any other object. Therefore, *Gai* fails to teach or suggest "associating two or more access control lists with a given filesystem object in a heterogeneous filesystem," as is claimed in claim 1.

Applicant agrees with the Examiner that *Gai* does not teach "a heterogeneous filesystem, wherein the heterogeneous filesystem comprises two or more differing types of filesystems," as is claimed in claim 1. However, the Examiner states that *Bradley* discloses this feature. The Examiner cites to the Abstract of *Bradley*:

> A translation system and method for translating file systems between nodes having heterogeneous file systems are provided. The translation system includes a consumer node having a first file system and a driver for supplementing requests from the first file system to a storage device. Also included in the translation system is an input/output (I/O) node which implements a second file system. The I/O node is connected to the storage device and is in communication with the consumer node over a transport. The I/O node includes a translator layer designed to map the supplemented requests from the first file system to the second file system and back to the first file system.

Bradley, abstract.

Here, *Bradley* describes a translation system for translating filesystems between nodes having heterogeneous file systems. Although *Bradley* does teach heterogeneous file systems, *Bradley* does not teach or suggest associating two or more access control lists with a given file system object in a heterogeneous filesystem in this or any other section of the reference.

The Examiner cites to the following:

> The method begins at an operation 102 wherein a set of file systems is provided. Exemplary file systems are those associated with operating systems such as, Unix.TM., Sun Microsystems Inc. Solaris.TM., Microsoft Corp. Windows.TM., Apple Computer Inc. Mac OS.TM., etc. These file systems include, for example, FAT16, FAT32, NTFS, HPFS, UFS, EFS, Berkeley FS, AT&T Unix FS, LFS, AFS, Unicos FS, etc. The method then moves to an operation 104 where metadata of each of the provided file systems is examined. This examination begins by operation 104a wherein a first metadata type is identified.

Bradley, column 7, lines 30-40.

This cited portion of *Bradley* teaches file systems associated with operating systems and examples of different filesystems. However, *Bradley* does not teach or suggest "associating two or more access control lists with a given filesystem object in a heterogeneous filesystem. In fact, *Bradley* does not even mention access control lists anywhere in the reference.

The Examiner also cites to *Bradley* at column 8, lines 1-24 which states:

> Following operation 104c, the method moves to operation 104d where it is decided whether there is more metadata to examine. As long as metadata exists whose type has not yet been determined, the method returns to operation 104a. However, once all the metadata types have been determined, the method moves onto operation 106 where the meanings and behavior of the file system metadata are abstracted. This operation is then followed by operation 106 where a dynamic flat file system is created through the generation of at least a directory class and a file class. That is, in operation 106, files, directories, and hard disk volumes are abstracted by objects. Thereafter, in operation 108, the dynamic flat file system is created by mapping of file metadata to file class and directory class attributes.

FIG. 2A depicts a directory class 202, a file class 204, and a volume class 206 of an object-based dynamic flat file system, in accordance with one embodiment of the present invention. Classes, acting as templates, instruct a compiler or other tools to construct objects, which are instances of objects of a class. Thus, as classes, directory class 202, file class 204, and volume class 206 specify attributes and operations of directory type objects, file type objects, and volume type objects of dynamic flat file system.

The first paragraph of the citation states the files, directories, and hard disk volumes are abstracted by objects. However, such disclosure is insufficient to teach or suggest associating access control lists with a filesystem object, particularly in light of the fact that *Bradley* does not even discuss access control lists in this or any other part of the reference.

The second paragraph describes classes that instruct a compiler to construct objects. Again, these teachings are insufficient to disclose or suggest associating access control lists with a file system object. Therefore, *Bradley* fails to make up for the deficiencies of *Gai*. Neither *Gai* nor *Bradley,* either alone or in combination, teach or suggest the feature of "associating two or more access control lists with a given filesystem object in a heterogeneous filesystem, wherein the heterogeneous filesystem comprises two or more differing types of filesystems," as is claimed in claim 1. Accordingly, the Examiner fails to state a *prima facie* obviousness rejection of claim 1 or any other claim in this grouping of claims because the cited references fail to teach or suggest all of the features of claim 1.

A.1.ii. The proposed combination of references, considered as a whole, does not teach or suggest the feature of "responsive to receiving, from a requestor, a request for an access control list associated with the given filesystem object, determining a filesystem type of the requestor," as in claim 1.

Additionally, the proposed combination of references, considered as a whole, does not teach or suggest the feature of, "responsive to receiving, from a requestor, a request for an access control list associated with the given filesystem object, determining a filesystem type of the requestor," as in claim 1. The Examiner asserts otherwise, referring to the following portion of *Gai*:

With a single, unified ACL defined per interface per direction and stored in a CAM-type memory, the intermediate network device is able to rapidly evaluate network messages. In particular, upon receipt of a packet at a first interface, a forwarding

entity at the intermediate network device tests the packet against the single, unified ACL stored in the corresponding portion of the CAM.

Gai at column 4, lines 26-32.

The cited portion of *Gai* teaches that an intermediate network device, such as a router, uses a single, unified access control list to rapidly evaluate network messages. A forwarding entity tests the received packet against the unified access control list. Although *Gai* mentions receiving a data packet over a network by an intermediate network device and testing the packet against a single access control list, **testing the packet against the single unified access control list** cannot be interpreted explicitly or implicitly as teaching or suggesting determining a filesystem type of a requestor. In fact, it is presumed that, because the intermediate device is routing packets from senders to recipients, the packet is being tested against an access control list generated by the intermediate device without regard for the filesystem type of the requestor, as the intermediate device will likely be forwarding the packet to the intended recipient on the network, rather than sending the packet back to the original sender. Thus, *Gai* does not teach or suggest **determining a filesystem type of the requestor**. In fact, as discussed above, *Gai* does not teach, suggest, or even mention filesystem types in this or any other section of the reference.

The Examiner cites to *Gai* at column 7, lines 24-34 which states:

> Each row of the ACL, such as ACL 416a, corresponds to an Access Control Entry (ACE) statement, such as ACE statements 502-514, which specify the various criteria for the ACL 416a. The columns of the ACL represent the specific criteria with which network messages are compared. For example, ACLs 416a-416d each include a separate column for source address 516, destination address 518, source port 520, destination port 522 and protocol 524. Those skilled in the art will understand that greater or fewer message criteria may be employed.

This portion of *Gai* describes the rows and columns of an access control list. Although the columns of the access control list may represent criteria by which network messages are compared, such teachings cannot explicitly or impliedly teach or suggest determining a filesystem type of a requestor in response to receiving a request for an access control list associated with the file system object. Again, *Gai* does not teach, suggest, or even mention a filesystem, a filesystem object, or access control lists associated with a filesystem object.

The Examiner cites to column 8, lines 9-15, which is quoted above. Although *Gai* discusses receiving messages by the network device and comparing the packet to access

control entry statements of an access control list in an intermediate network device, such teachings do not address filesystem types or access control lists associated with a given filesystem object. Thus, the teachings in this section of *Gai* are also insufficient to disclose or suggest determining a filesystem type of a requestor in response to receiving a request for an access control list associated with the given filesystem object.

Finally, the Examiner cites to *Gai* at column 5, lines 13-21, which states:

> Software entities (not shown) executing on the various end stations 306-312 and servers 313 and 314 typically communicate with each other by exchanging discrete packets or frames according to predefined protocols, such as the Transmission Control Protocol/Internet Protocol (TCP/IP), the Internet Packet Exchange (IPX) protocol, the AppleTalk protocol, the DECNet protocol or NetBIOS Extended User Interface (NetBEUI).

Here, *Gai* describes software entities communicating by exchanging packets using various protocols, such as TCP/IP protocol. The teachings of *Gai* merely describe protocols for exchanging packets over a network. The packets describes by *Gai* are not described as containing requests for access control lists associated with filesystem objects. Moreover, the protocols are not described by *Gai* as doing anything other than protocols for exchanging discrete packets or frames. *Gai* does not provide any teachings or suggestions for determining a filesystem type of a requester in response to receiving packets or frames. Nor does *Gai* teach or suggest that the packets could be a request for an access control list associated with a filesystem object. Thus, *Gai* does not teach, suggest, or mention determining a filesystem type of the requestor or receiving a request for an access control list associated with the filesystem object. Therefore, *Gai* fails to teach or suggest all of the features in claim 1.

As shown above, *Gai* is devoid of disclosure with regard to this claimed feature. The Examiner appears to agree that *Bradley* does not teach this claimed feature as the Examiner has not cited to any portion of *Bradley* as disclosing this feature. And in fact, *Bradley* does not teach or suggest this feature. Because *Gai* and *Bradley* do not teach or suggest all of the features of claim 1, the proposed combination of *Gai* and *Bradley,* when considered as a whole, does not teach or suggest all of the features of claim 1. Accordingly, the Examiner fails to state a *prima facie* obviousness rejection of claim 1 or any other claim in this grouping of claims.

A.1.iii. The proposed combination of references, considered as a whole, does not teach or suggest the feature of "returning an access control list from the two or more access control lists for the given filesystem object," as in claim 1.

Furthermore, the proposed combination of references, considered as a whole, do not teach or suggest the feature of "returning an access control list from the two or more access control lists for the given filesystem object," as in claim 1. The Examiner alleges this feature is taught by *Gai* at column 8, lines 14-15 which is quoted above. This portion of *Gai* only discloses that once a match is located between the network message and the access control entry of an access control list, the corresponding action is returned. As discussed above, the matching is performed in an intermediate network device, such as a router, rather than in a heterogeneous filesystem. Moreover, *Gai* does not teach, suggest, or even mention, two or more access control lists for a given filesystem object.

In addition, this section of *Gai* is discussing matching a network message with an access control entry, rather than **matching the filesystem type of the requestor with an access control list** from the two or more access control lists for the given filesystem object. In other words, *Gai* is not returning an access control list that matches a filesystem type of the requestor. Instead, *Gai* is performing an action if the message matches an access control entry statement **in an access control list**. *Gai* provides no teachings or suggestions as to what the action might be in this section of the reference. In particular, *Gai* does not teach or suggest that the action includes returning a matching access control list.

The Examiner also cites to *Bradley* at Figure 7 which illustrates:

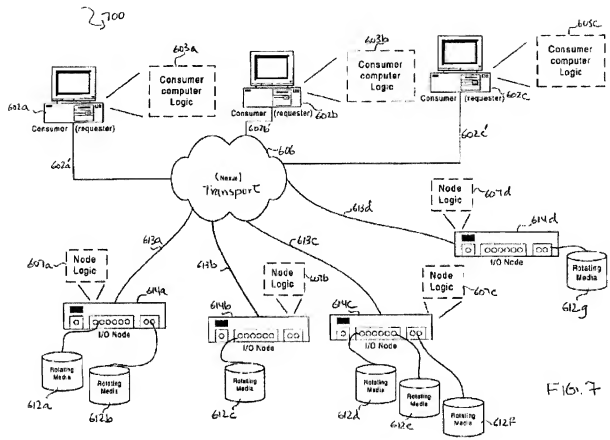
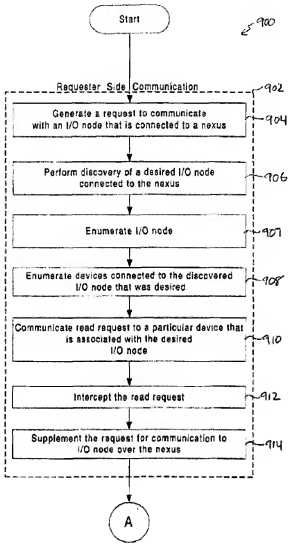
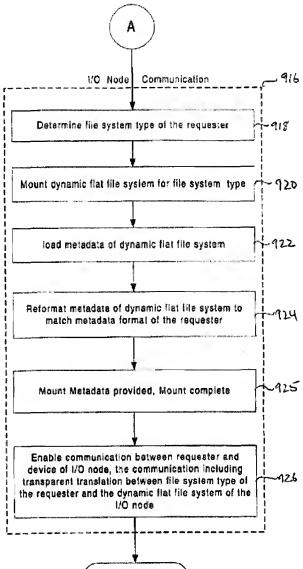


Figure 7 of *Bradley* shows an independent storage system including nodes having heterogeneous file systems. *See Bradley,* column 18, lines 1-5. However, Figure 7 does not show a request for an access control list associated with a filesystem object, a filesystem object associated with two or more access control lists, or returning an access control list from the two or more access control lists for the given filesystem object matching the filesystem type of the requestor.

The Examiner also cites to figure 9A of *Bradley* which illustrates:



Here, *Bradley* shows a flowchart illustrating a requester communicating with an I/O node. A request to communicate with the I/O node is generated. Figure 9B of *Bradley* illustrates as follows:



At step 918, the I/O node determines the filesystem type of the requester. *Bradley* describes this step in column 21, line 60 to column 22, line 5 which states:

> Subsequently, in an operation 916 of FIG. 9B, the I/O node communicates with a requester, in accordance with one embodiment of the present invention. This operation starts with an operation 918 where the I/O node determines the file system type of the requester. In one embodiment, this task is achieved as a result of the ISD layer intercepting and supplementing the consumer file system request.

That is, by intercepting and supplementing the issued request, the ISD layer informs the translator layer of the I/O node of the file system type of the requester. Thus, as of the time the ISD layer provides the type of the consumer file system to the I/O node, the I/O node associates that particular consumer with the communicated file system type.

As shown above, the I/O node of *Bradley* determines the filesystem type of the requester that is requesting communications with the I/O node. *Bradley* translates between the requester file system type and the dynamic flat file system of the I/O node at step 926. *Bradley* states "Thus, in an operation 926, the method enables communication between requester and a device of the I/O node by transparently translating between the requester file system type and the dynamic flat file system of the I/O node." *Bradley,* column 22, lines 31-34.

Although *Bradley* states that a filesystem type of a requester is determined, the requester is a requester requesting communications with an I/O node, rather than a requester requesting an access control list associated with the filesystem object. Because the requester is not requesting an access control list associated with the filesystem object, the requester of *Bradley* is not equivalent to the requester of claim 1.

Even if, *arguendo,* the requester of *Bradley* teaches or suggests the requester of claim 1, *Bradley* does not teach or suggest that **an access control list** from the two or more access control lists for the given filesystem object **matching the filesystem type of the requestor** is returned. In other words, *Bradley* only determines a filesystem type of the requester but does not return an access control list matching the filesystem type of the requester.

Moreover, *Bradley* teaches that the filesystem type of the requester is used to translate between the requester file system and the dynamic flat files system rather than **returning an access control list** ...matching the filesystem type of the requester. Thus, *Gai* and *Bradley,* either alone or in combination, fail to teach returning an access control list from the two or more access control lists for the given filesystem object matching the filesystem type of the requester.

Because *Gai* and *Bradley* do not teach or suggest all of the features of claim 1, the proposed combination of *Gai* and *Bradley,* when considered as a whole, does not teach or suggest all of the features of claim 1. Accordingly, the Examiner fails to state a *prima facie* obviousness rejection of claim 1 or any other claim in this grouping of claims.

A.1.iv. The Examiner fails to present a *prima facie* case of obviousness because the Examiner has not stated a proper reason to combine the references.

Additionally, the Examiner failed to state a *prima facie* obviousness rejection against claim 1 because the Examiner failed to state a proper reason to combine the references under the standards of *KSR Int'l.* As shown above, *Gai* and *Bradley* simply do not teach or suggest what the Examiner believes these references to teach and suggest. Therefore, the reasoning provided by the Examiner to combine the references rests on inherently flawed reasoning. For this reason, the Examiner did not state a proper, rational reason to combine the references as required by *KSR Int'l.* Accordingly, the Examiner failed to state a *prima facie* obviousness rejection against claim 1 or any other claim in this grouping of claims.

A.1.v. *Gai* teaches away from the invention in claim 1

Additionally, *Gai* teaches away from the invention in claim 1. As shown above, *Gai* teaches a method for organizing, storing, and evaluating access control lists in an intermediate network device, such as a router. *Gai* teaches using an ACL converter and a boolean manipulation engine to generate a single unified ACL for a given interface. *Gai* states:

> The invention relates to a method and apparatus for efficiently organizing, storing and evaluating access control lists (ACLs) for use by an intermediate network device of a computer network. The intermediate network device includes an ACL converter which, in turn, includes a boolean transformation engine that is operatively coupled to a boolean manipulation engine. The boolean transformation engine is configured to access the ACLs in first format and to translate them into a first boolean representation, such as binary decision diagram (BDD) format. The boolean manipulation engine is configured to perform one or more operations on the ACLs specified for a given interface, including a merge operation, so as to generate a single, unified ACL for the given interface. In order to resolve possibly conflicting actions output by the multiple ACLs, the ACL converter may utilize one or more predefined conflict resolution tables during the merging process. The boolean transformation engine then translates the single, unified ACL into a second boolean representation, such as a sum of products (SOP) format, and stores the single, unified ACL in a preferred memory structure for subsequent evaluation by the intermediate device.

Gai, abstract.

Gai uses a unified access control list or translates a first format into a boolean representation rather than determining a filesystem type of the requester and returning an

access control list from two or more access control lists for a given filesystem object matching the filesystem type of the requester. In other words, rather than returning the access control list matching the requesters access control list, *Gai* transforms or merges access control lists to create a different access control list format or unified access control list to test network messages. *See* column 8, lines 8-15, shown above. *Gai* also teaches away from the invention in claim 1 where *Gai* teaches a method in an intermediate network device, such as a router, rather than a method in a heterogeneous filesystem.

A.1.vi. *Bradley* teaches away from the invention in claim 1

Bradley teaches away from the invention in claim 1 where *Bradley* teaches one of ordinary skill in the art to use a translation system to translate commands communicated in a filesystem native to the consumer to a filesystem format of an input/output node rather than associating two or more access control lists with a filesystem object and returning the access control list from the two or more access control lists for the given filesystem object matching the filesystem type of the requestor. *Bradley* teaches at column 3, lines 21-34:

> Broadly speaking, the present invention fills the aforementioned needs by providing a file system translator capable of enabling efficient and transparent communication between nodes having heterogeneous operating systems and associated file systems. In one embodiment, the file system translator of the present invention facilitates the simultaneous transparent access of heterogeneous platforms running different operating systems and associated file systems to a file system of a provider node, such as a storage node or a file server. The file system translator preferably includes a translator layer, which translates the commands communicated in the format of a file system native to the consumer to the file system format of an input/output node.

However, by returning an access control list from the two or more access control lists for the given filesystem object matching the filesystem type of the requester, as claimed in claim 1, the client is not required to perform any conversion or translation of access control list formats. In fact, *Bradley* does not even mention access control lists. Thus, *Bradley* teaches away from the features of claim 1 where *Bradley* teaches translating from a native filesystem format to a filesystem format of the input/output node rather than returning an access control list associated with a filesystem object matching the filesystem type of the requester.

A.1.vii. No reason exists to combine *Gai* and *Bradley*, when the references are considered as a whole, because *Gai* and *Bradley* address different problems

Furthermore, even if, *arguendo*, the combination of references taught the features of claim 1, both *Gai* and *Bradley* seek to provide solutions to completely dissimilar problems. *Gai* teaches generating a unified access control list or a binary decision diagram format for use by an intermediate network device, such as a router, so that multiple access control lists can be assigned to a single interface.

> Access control lists are primarily used to provide security. Thus, for a given interface, only a single list is evaluated per direction. For purposes of security, moreover, the lists are relatively short. Nevertheless, the evaluation of such lists by software modules can significantly degrade the intermediate device's performance (e.g., number of packets processed per second). This degradation in performance has been accepted mainly due to a lack of acceptable alternatives. It is proposed, however, to expand the use of access control lists for additional features besides just security decisions. For example, access control lists may also be used to determine whether a given packet should be encrypted and/or whether a particular quality of service (QoS) treatment should be applied. Accordingly, it is anticipated that multiple access control lists may be assigned to a single interface. As additional access control lists are defined and evaluated per packet, the reduction in performance will likely reach unacceptable levels. Accordingly, a need has arisen to optimize the creation and evaluation of multiple access control lists so as to maintain, if not improve, packet processing speeds. This is especially true as more and more internetworking functionality is being implemented in hardware circuitry to increase the speed and performance of internetworking devices.

Gai at column 3, lines 30-40.

Thus, *Gai* is directed towards the problem of intermediate network devices utilizing multiple access control lists at an interface. *Bradley* is directed towards the problem of translating between nodes having heterogeneous filesystems. *Bradley* translates commands communicated in the format of a native filesystem to the filesystem format of an input/output node. In addition, as discussed above, *Gai* is implemented in an intermediate network device while *Bradley* is implemented in a filesystem. Thus, great differences exist between *Gai* and *Bradley*. When *Gai* and *Bradley* are considered as a whole, no rational reason exists to combine these references to achieve the invention of claim 1 due to the great differences between the cited references. Accordingly, under the standards of *KSR Int'l.*, the Examiner has failed to state a *prima facie* obviousness rejection against claim 1 or any other claim in this grouping.

A.2. Claim 11

Claim 11 is representative of this grouping of claims. Claim 11 is as follows:

> 11. An apparatus for managing access control lists in a filesystem, the apparatus comprising:
>
> a filesystem, wherein the filesystem is a heterogeneous filesystem, and wherein the filesystem includes a plurality of differing filesystem types and a plurality of access mechanisms, and wherein each access mechanism of the plurality of access mechanisms is associated with a filesystem type; and
>
> a file storage, wherein the file storage has stored therein at least one filesystem object and wherein a given filesystem object within the at least one filesystem object has associated therewith two or more access control lists;
>
> wherein the filesystem, responsive to receiving from a requestor a request for an access control list associated with the given filesystem object, determines a filesystem type of the requestor and returns an access control list from the two or more access control lists for the filesystem object matching the filesystem type of the requestor.

Claim 11 recites subject matter that is discussed above with regard to claim 1. Therefore, claim 11 is distinguishable over *Gai* and *Bradley* for at least the same reasons set forth in section A.1 above. In addition, claim 11 recites additional features that are not taught or suggested by *Gai* and *Bradley*. Regarding claim 11 the Examiner states:

> Gai and Bradley further disclose a file storage (see Fig. 4, element 408), wherein the file storage has stored therein at least one filesystem object (see col. 7, lines 29-32) and wherein a given filesystem object within the at least one filesystem object has associated therewith two access control lists (see Fig 4. elements 416a-416e; col. 6, lines 15-18); wherein the filesystem, responsive to receiving from a requester a request for an access control list associated with the given filesystem object (see col. 4, lines 26-32; col. 7, lines 24-32); determines a filesystem type of the requester (see col. 5, lines 13-21; col. 7, lines 29-34; column 8, lines 9-15) and returns an access control list from the two or more access control lists for the given filesystem object matching the filesystem type of the requester (see Gai col. 8, lines 14-15, "once a match is located, the corresponding action is returned and processing stops") and (see Bradley Figures 7 and 9, col. 18, lines 1-25).

Office Action dated May 31, 2007, at page 7-8.

A.2i The proposed combination of references, considered as a whole, does not teach or suggest the feature of "a file storage, wherein the file storage has stored therein at least one filesystem object and wherein a given filesystem object within the at least one filesystem object has associated therewith two or more access control lists," as in claim 11.

Furthermore, the proposed combination of references, considered as a whole, does not teach or suggest the feature of "a file storage, wherein the file storage has stored therein at least one filesystem object and wherein a given filesystem object within the at least one filesystem object has associated therewith two or more access control lists," as in claim 11. The Examiner alleges this feature is taught by Gai at Figure 4, which is shown above. Figure 4 merely shows access control lists stored on a non-volatile random access memory (NV-RAM) on an intermediate network device. The NV-RAM is not a file storage on a filesystem. Moreover, even if the NV-RAM could, *arguendo,* teach or suggest a file storage on a filesystem, *Gai* only illustrates access control lists on the NV-RAM. *Gai* does not show a filesystem object, or two or more access control lists associated with a filesystem object stored on a file storage.

The Examiner cites to column 7, lines 29-32 which is quoted above. This section of *Gai* describes rows and columns in access control lists. *Gai* does not teach or suggest filesystem objects associated with access control lists or storing filesystem objects associated with two or more access control lists.

The Examiner also cites to column 6, lines 15-18 which is included in the section of *Gai* that states:

> CPU 406, which may be configured to run a plurality of executable functions, such as an encryption algorithm 412 and a logging function 414, is coupled to NVRAM 408, which may contain one or more text-based access control lists (ACLs) 416a-416e, such as ACLs 101, 202, 303, 404 and 505, and also to dynamic memory 409. Forwarding entity 404 may include a plurality of conventional sub-components configured to implement QoS treatments, such as a packet/frame classifier 420, a scheduler 422, a shaper entity 424, a marker entity 426, a dropper entity 428, and a queue selector/mapping entity 430. The forwarding entity 404 is also coupled to the CPU 406, the TCAM 410, which may be apportioned into segments 410a-e whereby each segment 410a-e corresponds to a particular interface 402a-e, and memory 411. As described below, the forwarding entity 404 is basically configured to forward or switch network messages among the various interfaces 402a-e of device 316.

Gai, column 6, lines 13-30.

Here, *Gai* describes an NV-Ram coupled to central processing unit. The NV-RAM contains access control lists. Once again, *Gai* does not teach or suggest a file system object stored on a file storage or a file system object associated with two or more access control lists. Thus, *Gai* fails to teach or suggest ""a file storage, wherein the file storage has stored therein at least one filesystem object and wherein a given filesystem object within the at least one

filesystem object has associated therewith two or more access control lists," as in claim 11.

Because *Gai* and *Bradley* do not teach or suggest all of the features of claim 11, the proposed combination of *Gai* and *Bradley,* when considered as a whole, does not teach or suggest all of the features of claim 11. Accordingly, the Examiner fails to state a *prima facie* obviousness rejection of claim 11.

A.3 Claims 3 and 13

Claim 3 and 13 is representative of this grouping of claims. Claim 3 is as follows:

> 3. The method of claim 2, wherein the step of returning the matching access control list includes accessing the matching access control list using an access mechanism associated with the filesystem type of the requestor.

The Examiner rejects claim 3 as obvious over *Gai* and *Bradley.* This rejection is clearly in error and should be overturned. Claim 3 depends from claim 1. Therefore, the Examiner failed to state a *prima facie* obviousness rejection against claim 3 for the reasons presented above in the response to the rejection to claim 1. Additionally, the proposed combination of references, considered as a whole, does not teach or suggest the feature of "accessing the matching access control list using an access method associated with the filesystem type of the requester," as in claim 3. The Examiner asserts otherwise, citing to *Gai* at column 8, lines 14-15, column 7, lines 24-27, column 5, lines 13-21, column 7, lines 29-34; and column 8, lines, which are all quoted and discussed above. As discussed above, *Gai* does not teach or suggest determining a filesystem type of the requester. Likewise, *Gai* also fails to teach or suggest an access mechanism associated with the filesystem type of the requester or accessing the matching access control list using the access mechanism associated with the filesystem type of the requester. Therefore, both *Gai* and *Bradley* fail to teach or suggest the features of claim 3. Accordingly, under the standards of *KSR Int'l.*, the Examiner failed to state a *prima facie* obviousness rejection against claim 3 or the remaining claims in this grouping of claims.

A.4 Claims 8 and 18

Claim 8 and 18 is representative of this grouping of claims. Claim 8 is as follows:

> 8. The method of claim 1, wherein the step of associating two or more access control lists with a given filesystem object includes storing the two or more access control lists in file storage with the given filesystem object.

The Examiner rejects claim 8 as obvious over *Gai* and *Bradley*. This rejection is in error and should be overturned. Claim 8 depends from claim 1. Therefore, the Examiner failed to state a *prima facie* obviousness rejection against claim 8 for the reasons presented above in the response to the rejection to claim 1. Additionally, the proposed combination of references, considered as a whole, does not teach or suggest the feature of "storing the two or more access control lists in file storage with the given filesystem object," as in claim 8. The Examiner asserts otherwise, citing to *Gai* at column 7, lines 16-32 which is quoted above. This section of *Gai* describes the rows and columns in an access control list. However, *Gai* does not teach or suggest a filesystem object or storing two or more access control lists with the filesystem object.

The Examiner cites to Figure 4, which is shown above. Figure 4 merely shows an intermediate network device having access control lists in NVRAM. Although this figure arguable shows access control lists stored in a memory or data storage device, Figure 4 does not show a filesystem object or storing the access control lists with a filesystem object.

The Examiner cites to column 6, lines 1-2 which states: "non-volatile random access memory (NVRAM) 408, dynamic memory 409 and at least one content addressable or associative memory 410." This section merely lists types of memory. The Examiner cites to column 6, lines 13-18 which states: "CPU 406, which may be configured to run a plurality of executable functions, such as an encryption algorithm 412 and a logging function 414, is coupled to NVRAM 408, which may contain one or more text-based access control lists (ACLs) 416a-416e, such as ACLs 101, 202, 303, 404 and 505, and also to dynamic memory 409." Here, *Gai* states that a central processing unit is coupled to NVRAM which contains access control lists. *Gai* does not teach storing access control lists **with the given filesystem object.**

The Examiner also cites to column 7, lines 60-66 which states:

> The text-based ACLs that are to be utilized at a given intermediate device are then downloaded to that device in a conventional manner and stored, preferably in non-volatile memory. In particular, the ACLs may be maintained in memory as ASCII text or in other formats. For example, ACLs 416a-416e may be downloaded to device 316 by the network administrator and stored at NVRAM 408.

Again, *Gai* describes access control lists in memory, but does not disclose storing the access control lists **with the given filesystem object.** As discussed above, *Gai* does not teach or suggest a filesystem or a filesystem object. Thus, *Gai* also fails to teach or storing the two or more access

control lists in file storage with the given filesystem object. Therefore, both *Gai* and *Bradley* fail to teach or suggest the features of claim 8. Accordingly, under the standards of *KSR Int'l.*, the Examiner failed to state a *prima facie* obviousness rejection against claim 8 or the remaining claims in this grouping of claims.

A.5 Claims 2, 10, and 12

In view of the arguments above, independent claims 1, 11, and 20 are in condition for allowance. Claims 2, 10, and 12 are dependent claims depending on independent claims 1 and 11, respectively. Consequently, dependent claims 2, 10, and 12 of the present invention also are allowable at least by virtue of their dependence upon allowable claims.

B.1. Claims 4-7, 9, 14-17, and 19

The second ground of rejection is whether the Examiner failed to state a *prima facie* obviousness rejection under 35 U.S.C. § 103 against claims 4-7, 9, 14-17, and 19 as allegedly being unpatentable over *Gai* in view of *Bradley* and in further view of *Hitz* et al., File Access Control in a Multi-Protocol File Server, U.S. Patent No. 6,457,130, dated September 24, 2002 (hereinafter referred to as "*Hitz*"). This rejection is in error and should be overturned.

B.1.i Claims 4 and 14

Claim 4 is representative of this group. Claim 4 is as follows:

> 4. The method of claim 2, further comprising:
> responsive to a determination that a matching access control list does not exist, providing a new access control list for the filesystem type of the requestor; and returning the new access control list.

With respect to claim 4, the Examiner states that:

> **Regarding claims 4 and 14**, Gai discloses further comprising: responsive to a determination that a matching access control list does not exist (see col. 8, lines 24-26; col. 7, lines 24-27, "If no ACE of the subject ACL matches the message, an implicit action located at the end of the ACL is typically returned"),
>
> Gai and Bradley do not disclose "providing a new access control list for the filesystem type of the requestor; and returning the new access control list."
>
> However, Hitz explicitly discloses responsive to a determination that a matching access control list does not exist (see col. 6, lines 1-2), providing a new access control list for the filesystem type of the requester 9see col. 8, lines 26-34,

new access control limits); and returning the new access control list (see col. 8, lines 12-16; col. 8, lines 35-40; col. 8-lines 60-62).

Therefore, it would have been obvious at the time the invention was made to a person of ordinary skill in the art to have combined Hitz's invention within Gai and Bradley to include the step of providing a new access control list for the filesystem type when a matching access control list does not exist of the requester and returning the new ACL. One of ordinary skill in the art would have been motivated for the purpose of enforcing file access control among client devices using multiple diverse access control models and multiple diverse file server protocols (see Hitz col. 2, lines 36-40).

Final Office Action dated May 31, 2007, p. 8-9.

As shown in Sections A.1-A.4 above, the combination of Gai and Bradley does not teach or suggest all features recited in independent claims 1, 11, and 20. As a result, the combination of *Gai* and *Bradley* does not teach or suggest all features recited in independent claims 1, 11, and 20 of the present invention. In view of the arguments above, independent claims 1, 11, and 20 are in condition for allowance. Claims 4-7, 9, 14-17, and 19 are dependent claims depending on independent claims 1, 11 and 20, respectively. Consequently, dependent claims 4-7, 9, 14-17, and 19 of the present invention also are allowable at least by virtue of their dependence upon allowable claims.

In addition, claims 4-7, 9, 14-17, and 19 recite additional combinations of features that are not taught or suggested by the cited references. The Examiner alleges that *Gai* discloses "responsive to a determination that a matching access control list does not exist at col. 8, lines 24-26 which states: "If no ACE of the subject ACL matches the message, an implicit action located at the end of the ACL is typically returned (e.g., permit or deny)". Here, *Gai* is comparing a network message to an access control entry of an access control list in an intermediate network device. A comparison of a network message or packet to an entry in an access control list is not equivalent to matching an access control list to a filesystem type of a requester. In addition, as discussed above, *Gai* does teach or even mention a filesystem or a filesystem type of the requester. Therefore, *Gai* cannot be interpreted so broadly as teaching or suggesting a determination that a matching access control list **from the two or more access control lists for the given file system object** exists.

The Examiner also cites to *Gai* at column 7, lines 24-27, which is quoted above. As discussed above, this portion of *Gai* merely rows and columns in an access control list. *Gai* does

not disclose that the access control list is associated with a filesystem object. Therefore, this section of *Gai* cannot teach or suggest determining that a matching access control list from the two or more access control lists for the given filesystem object exists. Thus, *Gai* fails to teach or suggest the feature "responsive to a determination that a matching access control list does not exist." Applicant agrees with the Examiner that *Gai* and *Bradley* do not teach "providing a new access control list for the filesystem type of the requester; and returning the new access control list." In addition, *Hitz* fails to make up for the deficiencies of *Gai* and *Bradley*.

The Examiner alleges *Hitz* discloses the feature "providing a new access control list for the filesystem type of the requester; and returning the new access control list", as claimed in claim 4. The Examiner cites to *Hitz* at column 6, lines 1-2 which states "However, the file server 110 can also receive a request 121 that does not match the security style for the target file 112." Here, *Hitz* discloses a file server receiving a request to access a file that does not match the security style of the target file. The matching here is matching the request to access the file with the security style of the file. The matching is not a matching of a filesystem type of the requester with an access control list from the two or more access control lists associated with a filesystem object. Thus, the matching of *Hitz* is distinctly different than the matching of claim 1 and claim 4. Therefore, this section of *Hitz* does not teach or suggest responsive to a determination that a matching access control list does not exist.

The Examiner cites to column 8, lines 26-34 of *Hitz* which recites:

> When the file 112 has its access control limits modified, the file server 110 sets the security style for the file 112 equal to a security style associated with the new access control limits. (This is limited by restrictions imposed by access control trees, described herein.) Thus, if a client device 120 sets a set of Unix Perms for the file 112, the security style for the file 112 is set to Unix security style. Similarly, if a client device 120 sets an NT ACL for the file 112, the security style for the file 112 is set to NT security style.

This section of *Hitz* describes the process of setting the security style of the file to equal the security style of access control limits for the file. Again, *Hitz* is not providing a new access control list or returning the new access control list. Rather, *Hitz* is setting a security style to match **modifications** in an existing access control list.

The Examiner also cites to *Hitz* at column 8, lines 12-16 which states "Each file 112 has its security style set by the file server 110 so that either (a) a request 121 to perform an operation

on the file 112, or (b) a request 121 to perform an operation that sets the access control limits for the file 112, produce expected results". This portion of *Hitz* only states that a request produces the expected results. *Hitz* does not disclose that the request creates a new access control list or returns the new access control list for the filesystem type of the requester.

The Examiner cites to *Hitz* at column 8, lines 35-40 which states:

> The file server 110 can receive a request 121 to read or view the access control limits for a file 112. Also, when the file server 110 receives a request 121 to make an incremental change to the access control limits for a file 112, it determines the current access control limits for the file 112 before making the incremental change.

Here, the file server determines current access control limits for a file when a request is received to change access control limits for the file. Determining access control limits is very different than providing a **new access control list** and returning the new access control list. The Examiner also cites to *Hitz* at column 8, lines 60-62 which states that "[p]referably, the file server 110 performs dynamic permission mapping, in which the file server 110 maps the NT ACL into a set of Unix Perms at the time the mapping is required." In this section, *Hitz* discloses permission mapping. However, *Hitz* does not teach or suggest a new access control list or returning the new access control, as claimed in claim 4. Therefore, *Gai, Bradley,* and *Hitz,* either or in combination, do not teach or suggest all of the features of claim 4. Accordingly, the Examiner fails to state a *prima facie* obviousness rejection of claim 4 or any other claims in this grouping of claims.

B.1.ii. The Examiner fails to present a *prima facie* case of obviousness because the Examiner has not stated a proper reason to combine the references.

Additionally, the Examiner failed to state a *prima facie* obviousness rejection against claim 1 because the Examiner failed to state a proper reason to combine the references under the standards of *KSR Int'l.* As shown above, *Gai, Bradley,* and *Hitz* simply do not teach or suggest what the Examiner believes these references to teach and suggest. Therefore, the reasoning provided by the Examiner to combine the references rests on inherently flawed reasoning. For this reason, the Examiner did not state a proper, rational reason to combine the references as required by *KSR Int'l.* Accordingly, the Examiner failed to state a *prima facie* obviousness rejection against claim 1 or any other claim in this grouping of claims.

B.1.iii. Claims 5-7, 9, 15-17, and 19

In view of the arguments above, independent claims 1, 11, and 20 and dependent claims 4 and 17 are in condition for allowance. Claims 4-7, 9, 14-17, and 19 are dependent claims depending on independent claims 1, 11 and 20, respectively. In addition, claims 5-7 depend from claim 4 and claims 15-17 depend from claim 14. Consequently, dependent claims 4-7, 9, 14-17, and 19 of the present invention also are allowable at least by virtue of their dependence upon allowable claims.

Therefore, claims 4-7, 9, 14-17 and 19 are not obvious in view of *Hitz* because the features believed to be disclosed by this cited reference are not present.

Accordingly, Appellant respectfully urges the Board of Patent Appeals and Interferences not to sustain the rejection of claims 3, 4, 8-11, 18, 19, 24, 25, and 29-32 as allegedly being unpatentable over *Gai* in view of *Bradley* and in further in view of *Hitz*.

/Mari Stewart/
Mari Stewart
Reg. No. 50,359
YEE & ASSOCIATES, P.C.
PO Box 802333
Dallas, TX 75380
(972) 385-8777

CLAIMS APPENDIX

The text of the claims involved in the appeal are:

1. A method for managing access control lists in a filesystem, the method comprising:

associating two or more access control lists with a given filesystem object in a heterogeneous filesystem, wherein the heterogeneous filesystem comprises two or more differing types of filesystems;

responsive to receiving, from a requestor, a request for an access control list associated with the given filesystem object, determining a filesystem type of the requestor; and

returning an access control list from the two or more access control lists for the given filesystem object matching the filesystem type of the requestor.

2. The method of claim 1, further comprising:

determining whether an access control list matching the filesystem type of the requestor exists; and

responsive to a determination that a matching access control list exists, returning the matching access control list.

3. The method of claim 2, wherein the step of returning the matching access control list includes accessing the matching access control list using an access mechanism associated with the filesystem type of the requestor.

4. The method of claim 2, further comprising:

responsive to a determination that a matching access control list does not exist, providing a

new access control list for the filesystem type of the requestor; and

returning the new access control list.

5. The method of claim 4, wherein the step of returning the new access control list includes accessing the new access control list using an access mechanism associated with the filesystem type of the requestor.

6. The method of claim 4, wherein the step of providing a new access control list for the filesystem type of the requestor includes translating an existing access control list to the filesystem type of the requestor.

7. The method of claim 4, wherein the step of providing a new access control list for the filesystem type of the requestor includes providing a default access control list for the filesystem type of the requestor based on rules associated with the filesystem.

8. The method of claim 1, wherein the step of associating two or more access control lists with a given filesystem object includes storing the two or more access control lists in file storage with the given filesystem object.

9. The method of claim 1, wherein the step of associating two or more access control lists with a given filesystem object includes storing a native access control list in file storage with the given filesystem object and storing one or more non-native access control lists in access control list storage separate from the file storage.

10. The method of claim 9, wherein an access control list storage is provided an for each directory, each filesystem, or for each portion of a file system.

11. An apparatus for managing access control lists in a filesystem, the apparatus comprising:

a filesystem, wherein the filesystem is a heterogeneous filesystem, and wherein the filesystem includes a plurality of differing filesystem types and a plurality of access mechanisms, and wherein each access mechanism of the plurality of access mechanisms is associated with a filesystem type; and

a file storage, wherein the file storage has stored therein at least one filesystem object and wherein a given filesystem object within the at least one filesystem object has associated therewith two or more access control lists;

wherein the filesystem, responsive to receiving from a requestor a request for an access control list associated with the given filesystem object, determines a filesystem type of the requestor and returns an access control list from the two or more access control lists for the filesystem object matching the filesystem type of the requestor.

12. The apparatus of claim 11, wherein the filesystem determines whether an access control list matching the filesystem type of the requestor exists and, responsive to a determination that a matching access control list exists, returns the matching access control list.

13. The apparatus of claim 12, wherein the filesystem accesses the matching access control list using an access mechanism within the plurality of access mechanisms associated with the

filesystem type of the requestor.

14. The apparatus of claim 12, wherein the filesystem, responsive to a determination that a matching access control list does not exist, provides a new access control list for the filesystem type of the requestor and returns the new access control list.

15. The apparatus of claim 14, wherein the filesystem accesses the new access control list using an access mechanism associated with the filesystem type of the requestor.

16. The apparatus of claim 14, wherein the filesystem translating an existing access control list to the filesystem type of the requestor to form the new access control list.

17. The apparatus of claim 14, wherein the filesystem provides a default access control list for the filesystem type of the requestor based on rules associated with the filesystem.

18. The apparatus of claim 11, wherein the filesystem stores the two or more access control lists in the file storage with the given filesystem object.

19. The apparatus of claim 11, wherein the filesystem stores a native access control list in the file storage with the given filesystem object, the apparatus further comprising:
an access control list storage, wherein the filesystem stores one or more non-native access control lists in access control list storage separate from the file storage.

20. A computer program product, in a computer readable recordable-type medium, for managing access control lists in a filesystem, the computer program product comprising:

instructions for associating two or more access control lists with a given filesystem object in a heterogeneous filesystem, wherein the heterogeneous filesystem comprises two or more differing types of filesystems;

instructions, responsive to receiving from a requestor a request for an access control list associated with the given filesystem object, for determining a filesystem type of the requestor; and

instructions for returning an access control list from the two or more access control lists for the given filesystem object matching the filesystem type of the requestor.

EVIDENCE APPENDIX

There is no evidence to be presented.

RELATED PROCEEDINGS APPENDIX

There are no related proceedings.